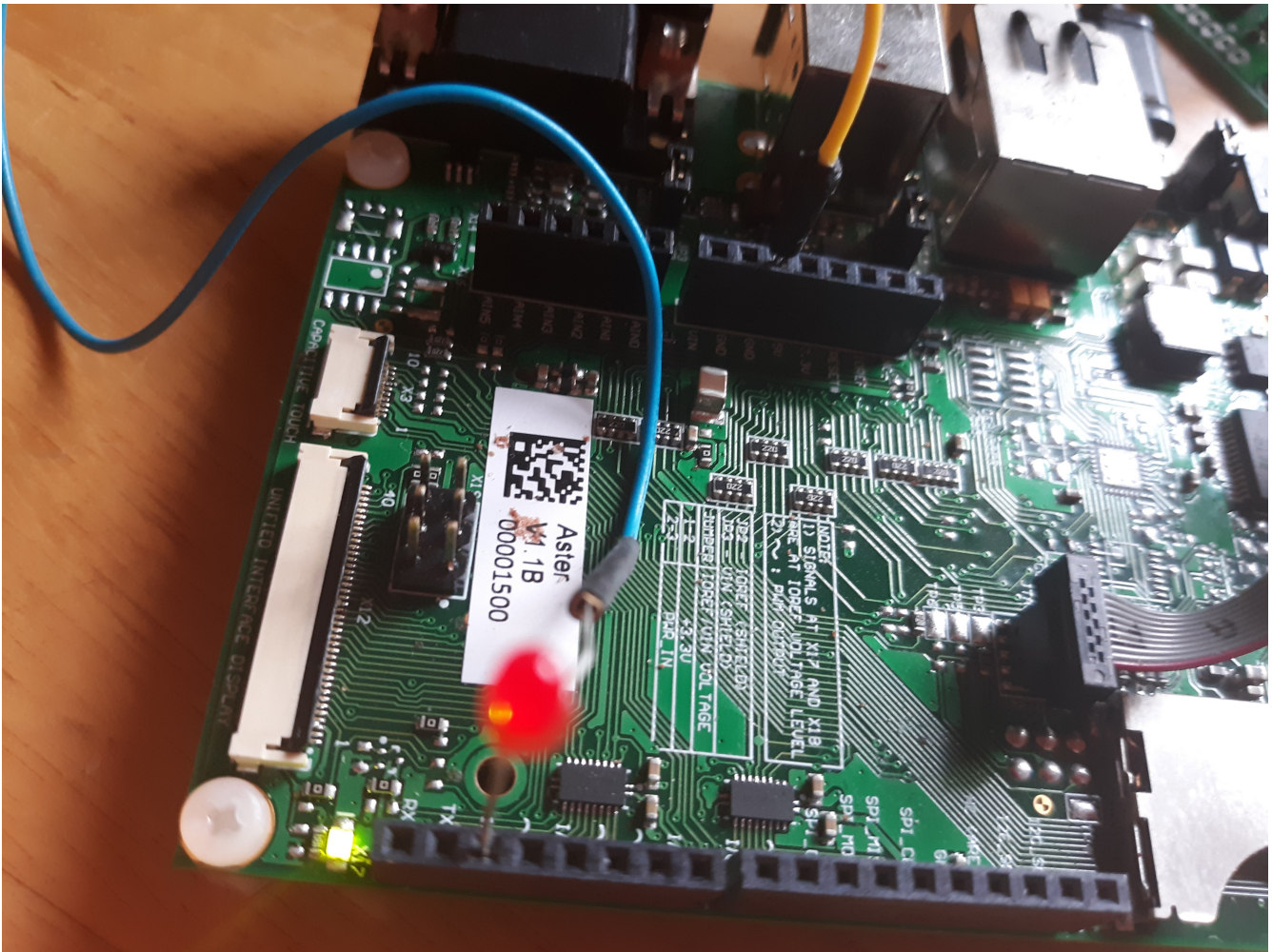




This is a Scalable PWM Modulator based on GPT on an Toradex Colibri imx7 module M4 core suitable for driving current through LED(s) or controlling brightness/contrast on a display and even perhaps motor control with some adjustments.

Make sure you get the pinning correct as I have changed the settings to allow PWM LED routing to the Arduino headers. If your LED burns, assume you routed incorrectly...Not assuming any responsibility for any damage that may happen to your board! With that out of the way, this one simple, fast approach to PWM control. A better, multi-channel controller/driver is on the way soon! Email me at : solaraeng@gmail.com if any issues or improvements you discovered...Enjoy!



/*

- * Copyright (c) 2015, Freescale Semiconductor, Inc.
- * All rights reserved.

- * Project - mx7_colibri_m4_PWM_imx_demo (PWM Modulator)
- * Created by : Mario Ghecea
- * Solara Engineering (solaraeng@gmail.com)
- * 6/29/2019
- * Purpose - To facilitate a scalable and programmable PWM algorithm within FreeRTOS
- * utilizing any number of dividing steps (1-n) for smoothness and PWM resolution
- * Only one generic timer (GPT) is used as counter for each alternating phase step...
- * This could be used as a generic LED driver, contrast for a display and perhaps
- * motor control through expansion.
- * If you reuse or distribute for your purpose please keep this header...

- * Redistribution and use in source and binary forms, with or without modification,

- * are permitted provided that the following conditions are met:
- *
- * o Redistributions of source code must retain the above copyright notice, this list
- * of conditions and the following disclaimer.
- *
- * o Redistributions in binary form must reproduce the above copyright notice, this
- * list of conditions and the following disclaimer in the documentation and/or
- * other materials provided with the distribution.
- *
- * o Neither the name of Freescale Semiconductor, Inc. nor the names of its
- * contributors may be used to endorse or promote products derived from this
- * software without specific prior written permission.
- *
- * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
- "AS IS" AND
- * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
- IMPLIED
- * WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
- * DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE
- LIABLE FOR
- * ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
- DAMAGES
- * (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR
- SERVICES;
- * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED
- AND ON
- * ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
- * (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF
- THIS
- * SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
- */

```
#include "FreeRTOS.h"
#include "task.h"
#include "board.h"
#include "debug_console_imx.h"
#include "gpio_ctrl.h"
#include "hw_timer.h"
```

```
#define PWM_MIN                (0.1f) // Lets assume 1% phase increases
#define PWM_MAX                (1.0f) // Lets assume your phase max at 100%
#define PWM_FREQ_DIVIDER      1    // Your PWM period frequency divider in milliseconds
(1/PWM_FREQ_DIVIDER) - LED Bink Interval
#define PWM_STEPS_PER_PHASE    20    // Increment PWM_STEPS_PER_PHASE
for a higher resolution (Above 20 it may exhibit some glitches)
#define PWM_RESOLUTION_COUNTER (1000/(PWM_FREQ_DIVIDER *
PWM_STEPS_PER_PHASE)) // Value in ms per phase
```

```

static volatile uint32_t blinkingIntervalHigh = PWM_RESOLUTION_COUNTER;
static volatile uint32_t blinkingIntervalLow = PWM_RESOLUTION_COUNTER;
static volatile float pwm = PWM_MIN;
static volatile bool start = true;

```

```

void SyncPWM();
void Resync();

```

```

/*****
*
* Function Name: ToggleTask
* Comments: this task is used to turn toggle on/off LED.
* This task has the effect of staring on cue so it will sync the phases
* correctly based on start signal from Phase Synchronizer (SwitchTask).
*
*****/

```

```

void ToggleTask(void *pvParameters)
{
    while (true)
    {
        if (blinkingIntervalHigh != 0 && start == true)
        {
            // If we have been signaled to start, lets sync the PWM phase
            SyncPWM();
            // Process PWM Phase High
            GPIO_Ctrl_ToggleLed(true);
            /* Use Hardware timer to get accurate delay */
            Hw_Timer_Delay(blinkingIntervalHigh);
            // Process PWM Phase Low
            GPIO_Ctrl_ToggleLed(false);
            Hw_Timer_Delay(blinkingIntervalLow);
        }
    }
}

```

```

// Sync PWM Block - Phase Increment
void SyncPWM()
{
    blinkingIntervalHigh = PWM_RESOLUTION_COUNTER * pwm ;
    blinkingIntervalLow = PWM_RESOLUTION_COUNTER * (1.0f - pwm);

    pwm += (0.5f / PWM_STEPS_PER_PHASE);

    Resync();
}

```

```

void Resync()
{
    if (pwm >= PWM_MAX)

```

```

    {
        pwm = PWM_MIN;
    }
}

```

```

/*****

```

```

*

```

```

* Function Name: main

```

```

* Comments: main function, toggle LED and switch the blinking frequency by key.

```

```

*

```

```

*****/

```

```

int main(void)

```

```

{

```

```

    /* Initialize board specified hardware. */

```

```

    hardware_init();

```

```

    Hw_Timer_Init();

```

```

    GPIO_Ctrl_Init();

```

```

    PRINTF("\n\r===== PWM Blinking Demo =====\n\r");

```

```

    /* Create a the APP main task. */

```

```

    xTaskCreate(ToggleTask, "Toggle Task", configMINIMAL_STACK_SIZE,
        NULL, tskIDLE_PRIORITY+1, NULL);

```

```

    /* Start FreeRTOS scheduler. */

```

```

    vTaskStartScheduler();

```

```

    /* should never reach this point. */

```

```

    while (true);

```

```

}

```

```

/*****

```

```

* EOF

```

```

*****/

```